Informed Access Network Selection: The Benefits of Socket Intents for Web Performance

Theresa Enghardt* and Philipp S. Tiesel* and Thomas Zinner* and Anja Feldmann‡
*TU Berlin, 10587 Berlin, Germany
‡Max Planck Institute for Informatics, 66123 Saarbrücken, Germany

Abstract—**Today's end-user devices have multiple access networks available and can achieve better application performance by distributing traffic across access networks. However, matching application traffic to the most suitable access network or bundling them is non-trivial, given varying application needs and network performance characteristics. Therefore, we propose an application-informed approach for access network selection (IANS). Based on the size of a Web resource, we select the better access network in terms of latency and available downstream capacity. We implement IANS within our Socket Intents prototype and evaluate its benefits for Web page loads under a variety of network conditions and for various Web pages. IANS provides the highest speedups for scenarios with asymmetric network conditions and for scenarios with low downstream capacity. Here, IANS improves relevant Web metrics by between 500 and 1000 ms in the median, compared to using the better of the two access networks, and may also outperform MPTCP. This confirms that IANS improves application performance over using a single network and, in several scenarios, even using MPTCP.**

I. Introduction

Most end-user devices today can connect to servers on the Internet via multiple access networks, e.g., WiFi and cellular. Therefore, such devices can choose between access networks or distribute traffic across them to improve application performance. However, applications have different needs, e.g., for short latency or high capacity. Thus, applications may benefit from choosing an access network with performance characteristics most suitable for their traffic. To choose the right access network, the device needs to know what to optimize for. Such knowledge is often available to applications, but it is often unknown to the network. To enable applications to share information about their expected traffic, we proposed Socket Intents [23]. Socket Intents enables informed access network selection (IANS) based on application information as well as current network characteristics. Hereby, IANS can distribute transfers, such as Web resource loads, according to their sizes: Loading small resources benefits from a network with short latency, while loading larger resources benefits from a network with high available downstream capacity.

In contrast, existing mechanisms for selecting an access network often apply simple, static rules. For example, most end-user devices default to WiFi if available and use cellular only as a fallback. However, using WiFi may lead to suboptimal application performance, as WiFi is not always better than cellular [6]. Furthermore, this strategy does not allow bundling the networks to aggregate their capacity. Multipath TCP (MPTCP) can combine multiple access networks for a single connection. Yet, prior work suggests that while MPTCP provides performance benefits for loading large resources, it may not speed up loading small resources [3]. As MPTCP splits all connections across networks in an application-agnostic way, it is not aware of individual resources, their sizes, and the requirements of the application. IANS aggregates access network capacity at a different granularity than MPTCP: Instead of splitting all connections across all access networks, IANS distributes individual resource loads. It does so in an informed manner, as IANS is aware of resource sizes as well as latency and available downstream capacity on each network. Therefore, IANS can load each resource over the most suitable access network. Moreover, IANS only needs client-sided support, while MPTCP needs both server- and client-sided support.

In this paper, we make the following contributions: 1) We apply IANS to Web browsing using our Socket Intents prototype [23]. We refine the Threshold Policy [7] to speed up loading Web pages "in the wild" as opposed to self-generated static Web pages hosted on a single server. In particular, we refine the Threshold Policy to better estimate available capacity on an access network and predict resource load times more accurately. 2) We perform a systematic study of the impact of IANS on Web performance in a wide range of network scenarios in a testbed, comparing it to using a single access network and MPTCP. We find that IANS significantly improves Web performance, e.g., reduces Above-The-Fold Times by between 500 and 1000 ms in the median, for scenarios with asymmetric network characteristics and for symmetric scenarios with low downstream capacity. In our asymmetric network scenarios, one access network has short latency but low capacity and the other has high capacity but large latency. Here, IANS often outperforms MPTCP, as MPTCP can experience performance problems caused by self-induced congestion on the low capacity network. 3) We confirm the achieved speedups for Web servers "in the wild". IANS has an advantage over MPTCP because MPTCP deployment is still limited, while IANS is available today because it does not need server-sided support.

II. Informed Access Network Selection

Our IANS is based on two key concepts: *Socket Intents* and *IANS Policies*. Based in the information provided by Socket Intents, IANS Policies decide which access network to use. Below we revisit both concepts:

978-3-903176-24-9 © 2019 IFIP

A. Socket Intents

Socket Intents [23], [27] are pieces of information that an application can provide about its needs and expectations. Intents are not QoS requirements or resource reservations, but optional hints that are available within the application or can be gathered with tenable effort.

In this paper, we focus on the SIZE TO BE RECEIVED Intent, which is the number of bytes that an application expects to receive in reply to a request. The application can learn this information from metadata, e.g., for resources embedded in a Web page, or it can query it, e.g., using HTTP HEAD or Range requests. Alternatively, the TRAFFIC CATEGORY Intent allows to optimize for, e.g., short latency or high downstream capacity. For loading Web resources, an application typically does not know what to optimize for, so we compute this information from the SIZE TO BE RECEIVED. For other Web use cases, such as uploading files, applications can set the SIZE TO BE SENT. By setting the COST PREFERENCES Intent, an application can balance potential performance improvements with cost, e.g., when using leftover capacity of a cellular data plan.

B. IANS Policies

IANS Policies decide how to distribute traffic across multiple access networks, see Figure 1. First, 1) an application specifies its Intents for a new connection or request, 2) an IANS Policy is queried, which decides which access network to use. Then, 3) the IANS Policy communicates this decision and 4) the new connection or request is scheduled on the chosen access network. Each IANS Policy can implement its own decision logic, taking as input the current network performance characteristics and the Socket Intents from the application.

For Web browsing, we build upon the THRESHOLD POLICY [7]. This IANS Policy is based on the idea that the load times of Web resources of different sizes are often dominated by latency or downstream capacity:

- For small resources, latency dominates: We choose the access network with the shortest latency available.
- For large resources, downstream capacity dominates: We calculate the expected load time for each available access network based on currently available capacity as well as latency and choose the network with the shortest estimate.

We name this IANS Policy the Threshold Policy based on the threshold resource size below which latency dominates and above which capacity dominates. By selecting the more suitable access network based on the SIZE TO BE RECEIVED Socket Intent, the Threshold Policy aims to reduce load times of individual resources, thus, to shorten the overall load time of the page. For details, we refer to Section III-C and to the source code of the Threshold Policy implementation within the Socket Intents prototype, see https://github.com/fg-inet/socket-intents/blob/release-0.8/policies/threshold_policy.c.

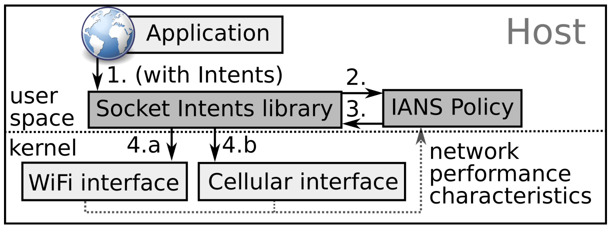


Fig. 1: IANS: Socket Intents / Policy interactions.

III. SOCKET INTENTS PROTOTYPE IMPLEMENTATION

We implement IANS within the Socket Intents prototype[1]. Our prototype allows an application to specify its Socket Intents on a per-request basis through the Socketconnect API, which we use within a Web proxy.

A. Socketconnect API

To enable applications to benefit from per-request IANS, the Socket Intents prototype implements the *Socketconnect* API[2]. Through this API, the application specifies the remote host name and port for each request, e.g., each Web resource, as well as a list of Socket Intents, such as the SIZE TO BE RECEIVED. The Socket Intents prototype then returns a socket, bound to a specific access network, to the application. Note that the prototype can also enable MPTCP for individual connections to use both networks[3]. Thus, when the application sends a request and receives the response, it uses the access network selected by IANS. In case the application has not previously communicated with this remote host, the socket corresponds to a newly established connection. Otherwise, the socket may correspond to a pre-existing connection, which the application can re-use. After finishing sending the request and receiving the response, the application can release the socket, at which point the socket becomes part of the pool of available connections managed within our prototype.

B. Web Proxy

Because browsers are complex, fast-moving systems and we want to be flexible to use different browsers with our prototype, we implement Socket Intents support in a Web proxy. While this allows us to study IANS, the proxy adds a slight overhead. To avoid overhead, future work should implement Socket Intents in the browser itself. Alternatively, IANS can be implemented in an HTTP library, enabling all applications that use this library to benefit from IANS without further modification.

For our study, the IANS-enabled Web proxy uses the Socketconnect API and provides the “Size to be Received” for each resource to be loaded. To determine the size of an individual resource, the proxy performs a two-step download: It first fetches the first 1000 Bytes using the Content-Range header via the shorter latency network. If the resource size is 1000 Bytes or smaller, the proxy receives the full resource with this first request. Otherwise, it receives the first part

[1]The code is publicly available at https://github.com/fg-inet/socket-intents.

[2]The current prototype supports HTTP/1.1, but support can also be added to, e.g., an interface to an HTTP/2 library or a QUIC implementation

[3]We leave IANS Policies using MPTCP as future work.

and knows the size of the rest of the resource from the Content-Length header[4]. Next, the proxy uses the Size to be Received Socket Intent so that the IANS Policy can select a suitable access network for loading the rest of the resource. Using the Socketconnect API allows the proxy to re-use TCP connections where appropriate, i.e., when loading another resource from the same remote host. Thus, the proxy can benefit from IANS on a per-request basis without imposing unnecessary delay for small resources.

C. Policy Implementation

The IANS Policy decides which access network to use for each request an application makes through the Socketconnect API, e.g., each Web resource. This decision is based on the Socket Intents of the request and on the current performance characteristics of the available access networks. While Socket Intents are specified by the application for each request, the prototype continuously queries estimates of current network performance characteristics kept by the host[5], see [7], and makes them available to the IANS Policy.

To select an access network for a given resource, the Threshold Policy predicts whether the resource load time is dominated by latency or available downstream capacity, see Section II-B. To do so, it compares two components of resource load time: It computes a latency component based on a two-way latency estimate and a capacity component based on an estimate of the available downstream capacity. As an estimate for the two-way latency of an access network, the Threshold Policy uses the Smoothed Round Trip Times (SRTTs) of the TCP connections which are currently established for each access network. To estimate the latency component of resource load time, the Threshold Policy has to multiply this two-way latency by the minimum number of necessary round trips to load the resource[6]. To compute an estimate of the available downstream capacity, the Threshold Policy assumes that the new transfer will get a fair share of the maximum capacity[7]. For the maximum capacity, the prototype observes download bitrate over time[8] and uses the maximum observed bitrate within the last 5 minutes as an estimate.

If the resource load is latency bound, the Threshold Policy selects the network with the shortest estimated latency. If the resource load is capacity bound, the Threshold Policy predicts total resource load time over all available networks. To do so, for new connections, is uses a model of TCP slow start rounds: In each round, a number of bytes of the resource is loaded, starting with the initial congestion window. This number doubles each round up until the available capacity is reached, at which point TCP slow start ends. The policy counts the number of rounds and multiplies them by the two-way latency estimate. To encourage connection reuse, the Threshold Policy gives reused connections an advantage by adding latency and capacity part instead of modeling slow start when calculating load time. After predicting load times of a resource over all networks, the Threshold Policy selects the network with the shortest expected load time.

[4]In case of missing Content-Length, the IANS Policy selects the shorter latency network by default.

[5]Having such performance estimates available requires traffic to be present on each network. To bootstrap our evaluation, we send a probing iperf flow over each network before starting to load any Web pages.

[6]This number depends on whether a new connection has to be established or an existing connection can be re-used.

[7]Based on the known concurrent TCP connections. To compensate for idle connections, the policy weights the number of concurrent connections by the current actual usage of the access network, which it computes based on the current and maximum bitrate.

[8]It periodically reads the network interface counters of all available network interfaces and computes the difference of these counters between subsequent reads.

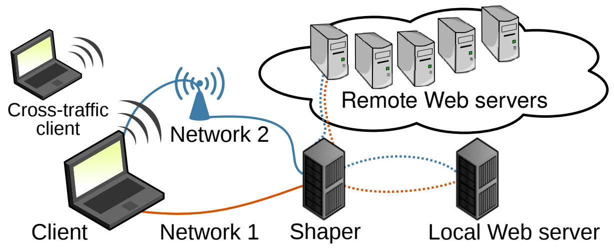


Fig. 2: Testbed setup.

IV. Evaluation Methodology

Next, we outline our methodology for studying the benefits of IANS for Web performance. To enable a systematic evaluation using network characteristics in which multiple access networks may improve performance, we set up a testbed with mirrored Web pages. In addition, we confirm the achievable speedups by accessing Web pages from servers "in the wild".

A. Network scenarios

We use a testbed, see Figure 2, which represents a scenario where the access network is the performance bottleneck and where we have full control over almost all components. It consists of a client, a traffic shaper, a Web server, and two network links, a wired and an 802.11 wireless link. To include the effects of actual WiFi we realize the wireless link via a WiFi Access Point (AP). Yet, to limit side effects we use a stationary AP on an otherwise unoccupied 5 GHz channel.

The client is connected to the traffic shaper via two access networks: To network 1 via a 1 Gbit/s Ethernet wired link with less than 1 ms delay, and to network 2 via the 802.11 wireless link. The wired link adds minimal delay and no congestion, as it directly connects the client to the shaper. The wireless link adds delay and congestion related to WiFi. With these two access networks, the client can either load the Web pages mirrored on the Web server via each of the networks or it can bundle the capacity of both networks using IANS or MPTCP.

For the systematic evaluation, we follow a factorial design approach which covers a wide range of network characteristics in terms of delay, downstream network capacity, and cross-traffic on the WiFi channel, see Table I. More specifically, delay and throughput restrictions are realized by the shaper. Note, the shaper clearly dominates the network characteristics of network 1 while network 2 also sees the effects of WiFi. We do not restrict upstream capacity, as it is not a limiting factor for loading Web pages.

TABLE I: Shaping levels

Property	**Levels**
Additional latency:	10, 50, or 100 ms.
Downstream capacity:	2, 5, 10, or 20 Mbit/s.
WiFi crosstraffic:	none, constant UDP, variable TCP.

Cross-traffic can be imposed on the WiFi network and is realized by using another client which sends traffic on the same channel. We impose both a constant load of cross-traffic which fully utilizes the WiFi using an iperf UDP flow and a self-similar load of TCP flows to fully utilize the shaper link using Harpoon [26][9].

To evaluate the achievable speedups "in the wild" the client can reach the Internet via the traffic shaper. Our vantage point is located in a well-connected university network. Thus, when using the above shaping parameters the access networks should remain the bottleneck. To confirm this we repeatedly (5 times) downloaded our chosen Web pages without any traffic shaping and confirm that the median/75th quantile latencies are lower at 17/26 ms than most shaper latencies and the receive times for large resources ($>$ 320 KB, the 99th quantile of the resource sizes) are small with a median of 41 ms. Thus, the network conditions imposed by the traffic shaper have a significant impact on the performance of "in the wild" Web page downloads. To minimize side-effects due to DNS, we run a resolver on the traffic shaper. For each domain name, this resolver returns the same set of IP addresses in the same order. It caches the results for the experiment duration of 20 hours. Thus, within each experiment, subsequent page loads are likely to be directed to the same CDN node. Furthermore, we randomize the order of page loads with different access selection policies to further limit the impact of Internet performance variations on any particular policy.

B. Web Workload

Our goal is to evaluate IANS vs. traditional access selection policies across a diverse set of Web pages popular among actual users while not falling into typical pitfalls, e.g., initial redirects [8]. We base our workload on the Alexa top list. To limit bias [22] we selected 102 highly ranked Web sites from nine categories[10]. This resulted in a list of domain names which typically redirect to some landing page. Since such initial redirects inflate load times [8] we eliminated redirects by manually loading each page once and then adding the resulting URL to our hit list. For some sites, mainly social media sites, we find that the resulting URL does not reflect actual user experience as the landing page we saw consisted of only a few objects. For those, we picked an alternative publicly accessible subpage of the same site, e.g., a publicly accessible social media profile.

[9]We configured Harpoon to generate TCP flows with an average total throughput similar to the shaped downstream capacity, whereby file sizes follow a Pareto distribution with alpha=1.2 and shape=1500 bytes and interconnection times follow an exponential distribution with a mean of one second.

[10]Our categories are: Search engines, News pages, shopping, social media, sports, arts and entertainment, business, games, and science.

Another difficulty we encountered is that the content of Web pages changes frequently, even across consecutive page loads. Thus, we mirror all 100 Web pages in our workload[11] to the Web server in our testbed[12], see Section IV-A. This setup allows us to fetch the same version for all these pages across experiments. When accessing Web pages "in the wild", we cannot enforce that the same version of a page is retrieved. Thus, to ensure comparability of the results we only include those Web page loads in our evaluation where the resulting resource count and page size differ by less than 1% within the same run.

We opt for hosting Web pages on a dedicated server instead of using an emulator, such as Mahimahi [18], which replays Web page loads on different emulated network conditions on a single host. Our reasons are twofold: First, our testbed enables a wider range of scenarios, e.g., realistic access network conditions using an actual WiFi Access Point and cross-traffic. Second, we can compare loads of the mirrored version to Web page loads "in the wild" given similar network conditions.

C. Web Performance Metrics

We capture the following Web performance metrics: 1) The Page Load Time (PLT) between the navigationStart and the onLoad event via the Navigation Timings API. 2) The Above-The-Fold Time (ATF) to load all user-visible content as computed by the browser plugin by da Hora et al. [5]. 3) The Mean Opinion Score (MOS) using Byte Index until ATF based on the WQL model [13]: $MOS = -0.4731 * ln(ByteIndex_{ATF}) + 7.0813$. Byte Index is the integral of resource sizes over resource load times, see Bocchi et al. [2]. We learn the resource sizes from the Content-Length header, or, if not present, the body size, as exported in the HTTP Archive (HAR) file after each page load.

D. Web Page Loads

For our evaluation, we repeatedly load Web pages using different access selection policies. We instrument Firefox 63.0.4 via the Marionette browser automation interface. To prevent skewed results due to browser caching, we set up a fresh browser profile for each page load. To allow the browser to leverage IANS, it uses a local Web proxy[13] as discussed in Section III-B. We compare the following cases: Loading a page using only a single network, using both networks with MPTCP with the primary subflow on the shorter latency network, and using Socket Intents with its "Threshold policy",

[11]We mirrored pages on January 21, 2019. We load each page once, store all resources locally, and copy them to our Web server. For each origin, i.e., each host that contributes content to the original page, we create a virtual host on our server.

[12]Our Web server is equipped with 8 cores and 16 GB of RAM, similar to a 2xlarge Amazon EC2 instance and runs an Apache Web server.

[13]We evaluate the overhead of the proxy by comparing PLTs with and without the proxy. We load Web pages on network 1 without traffic shaping. PLTs with proxy increases by a median of 6.3%. There are two main reasons for this increase. First, the browser supports TLS False Start, but OpenSSL, which the proxy uses, does not. Thus, the browser has one fewer round trip for each TLS handshake than the proxy. Second, the Socket Intents-enabled proxy uses a two-step download for querying the "Size To Be Received".

see Section II-B. We repeat each page load 5 times to eliminate measurement artifacts.

V. Evaluation Results

Next, we report on the benefits of IANS in a single scenario and in our systematic study, in which we highlight under which network conditions it yields which benefits.

A. *Asymmetric network scenario: In-depth discussion*

We start our exploration by focusing on the asymmetric network scenario. Our motivation for starting with this case is that here IANS should yield the biggest benefits for Web performance. In particular, network 1 has a short latency of only 10 ms but a limited downstream capacity of 2 Mbit/s. Network 2 has a downstream capacity of 20 Mbit/s but a latency of 100 ms.

Figure 3 shows the Web performance of all access selection policies according to two metrics, Page Load Times (PLT) and Above-The-Fold Times (ATF), as empirical cumulative distribution (ECDF) across all 100 Web pages.

IANS outperforms all other access selection policies for ATF and for almost all cases for PLT. Only MPTCP can lead to shorter PLTs, see Figure 3a. When investigating the cause, we find that IANS, as intended, benefits from the asymmetric network characteristics: Small resources benefit from the short latencies of network 1, while large resources benefit from the high downstream capacity of network 2. Since MPTCP distributes all resource loads across both networks, it saturates the short latency network first. This leads to congestion on this network which, in turn, inflates the load times of smaller resources. Thus, ATFs and PLTs increase for MPTCP. IANS avoids unnecessary congestion on the short latency network since it mainly uses the high downstream capacity network for retrieving large resources. Furthermore, IANS yields a statistically significant improvement in MOS (plot not shown) with a mean MOS difference of 0.11 compared to using the "better" (lower PLT) of the two single networks, and of 0.07 compared to using MPTCP.

Using a single network (either network 1 or network 2) leads to the worst PLTs and ATFs with a huge spread ranging from 0.7 seconds to more than 53 seconds. Using the better of the two networks avoids some of the outliers; none of the page loads takes longer than 20.4 seconds. Thus, in this scenario, there is no single "best" network to use.

Adding cross-traffic to the WiFi network 2 results in slightly longer load times but does not change the general observations (plots not shown). Both IANS and MPTCP are able to cope with the cross-traffic. The speedups vs. using a single access network are comparable to the scenario without cross-traffic. When adding variable cross-traffic to network 1, which has a short latency but low capacity, IANS still outperforms MPTCP. However, as both IANS and MPTCP are sensitive to congestion on the lower latency network, using only network 2, which does not have cross-traffic, becomes the better option. Therefore, a future version of IANS should detect high congestion on a network and then avoid loading resources over this network.

B. *Systematic Study of Scenarios*

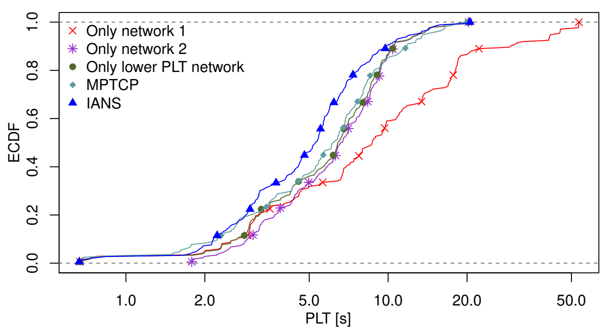
Next, we look at the results of the systematic study of 33 network scenarios with latencies of 10, 50, or 100 ms and downstream capacities of 2, 5, 10, or 20 Mbit/s for both access networks. Rather than using 33 ECDFs to visualize the results, we show, using heatmaps, the median ATF improvement of IANS and MPTCP vs. the "best" single network (according to PLT), see Figure 4. Note, the results for PLT and MOS show similar results (plots omitted). Each subplot focuses on a different scenario, namely, Figures 4a and 4d on access networks with symmetric network conditions, Figure 4b and 4e on access networks with asymmetric conditions where network 1's do not change and Figure 4c and 4f on access networks with asymmetric conditions where network 2's do not change. The color schema (same for all plots) goes from green for large performance improvements over light yellow for no significant differences to violet for large performance penalties. Each heatmap entry contains the median ATF improvement in ms and the corresponding confidence intervals.

Overall, green which corresponds to significant improvements dominates the results for IANS and MPTCP. However, red and violet are more common for MPTCP. Yellow dominates if the downstream capacity is large for both networks. Note, that the size of the confidence intervals can be large, i.e., hundreds of milliseconds, as load times fluctuate between page loads. Some of these effects are due to influences unrelated to networking, such as in-browser processing, and different orders in which the browser may load the resources. Other effects are related to the multitude of different Web pages and that the Web page load times range from less than a second to more than 20 seconds.

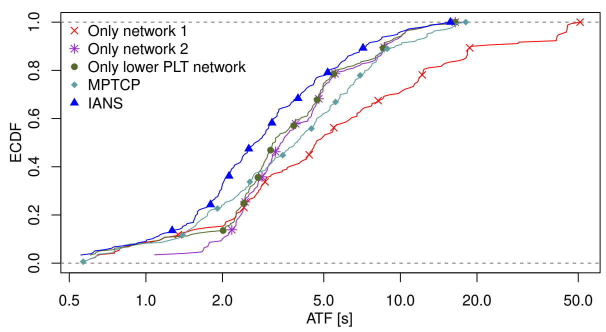
IANS shows the largest speedups in asymmetric scenarios, see Figure 4b and 4c, similar to the one discussed Section V-A. For MPTCP, this is not always the case, see Figure 4e and 4f. The reason is that IANS (for the 10/100 ms latency case) in contrast to MPTCP can indeed load small/large resource over the short latency/high capacity network while MPTCP tends to use both and, thus, may suffer from head-of-line blocking and congestion on the low capacity network.

Even for less asymmetric scenarios (10/50 ms, 50/100 ms latency) IANS still improves ATFs over MPTCP, see Figures 4b and 4e. MPTCP saturates network 1 (the lower latency one) first, which, again, leads to congestion on network 1 and inefficient use of the high capacity of network 2. MPTCP still degrades ATF if both networks have the same short latency (10ms), but asymmetric downstream capacities (2/10 Mbit/s or 2/20 Mbit/s), see Figure 4e. Since network 2 is the WiFi network, it adds delay and, thus, MPTCP will again saturate the lower latency one rather than taking advantage of the higher capacity one.

For symmetric scenarios with low downstream capacities IANS again shows significant ATF improvements, see Figure 4a. Here, both networks should be used, so IANS's capability of distributing resource across the networks improves resource load times and, thus, PLT and ATF. Speedups increase

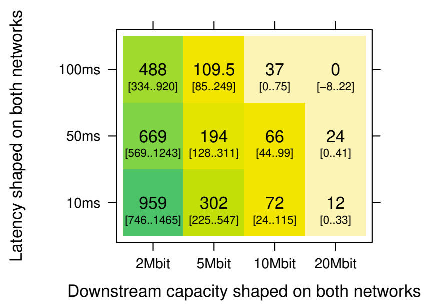


(a) ECDF of Page Load Times (PLT)

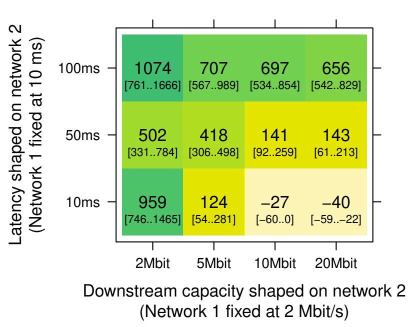


(b) ECDF of Above-The-Fold Times (ATF)

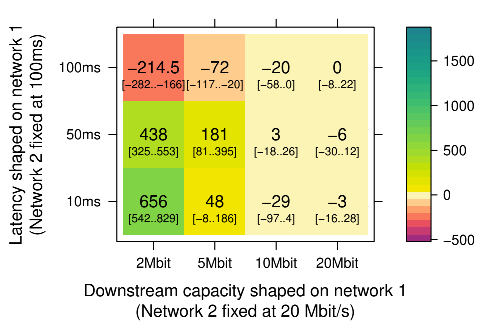
Fig. 3: Asymmetric scenario (network 1: 10 ms, 2 Mbit/s, network 2: 100ms, 20 Mbit/s)



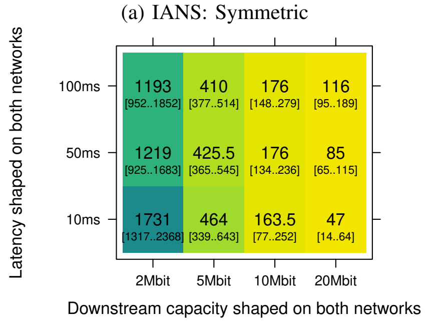
(a) IANS: Symmetric



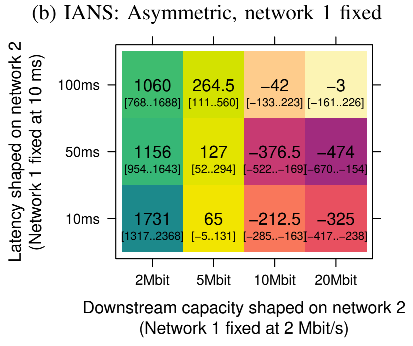
(b) IANS: Asymmetric, network 1 fixed



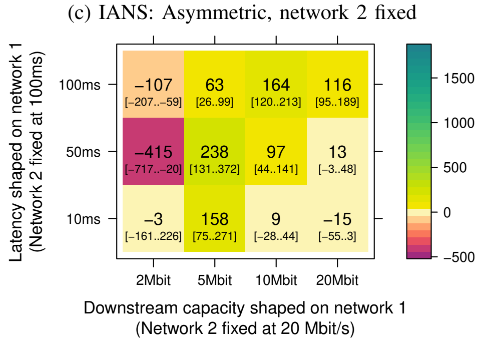
(c) IANS: Asymmetric, network 2 fixed



(d) MPTCP: Symmetric



(e) MPTCP: Asymmetric, network 1 fixed



(f) MPTCP: Asymmetric, network 2 fixed

Fig. 4: ATF improvements vs. using the single "better" network (median [ms] plus confidence intervals).

as latencies decrease due to less per-connection overhead. For purely symmetric scenarios, MPTCP outperforms IANS, see Figure 4d, as MPTCP is able to use the bundled network capacity at a finer granularity by using subflows. Thus, if a page's resources are skewed, i.e., have one large resource, MPTCP can efficiently use both networks while IANS cannot.

In scenarios with high downstream capacity, e.g., Figure 4a and 4d neither IANS nor MPTCP yields major benefits as a single access network already provides sufficient network resources. Still, MPTCP is slightly better than IANS since it is able to reuse all TCP connections. However, future versions of IANS can detect such network conditions and then default to MPTCP. With cross-traffic on network 2, using MPTCP remains the best option in most cases. However, using only network 1, which is not impacted by cross-traffic, is even better in case network 1 has sufficient capacity. Therefore, enabling MPTCP only selectively using IANS may have an advantage over using MPTCP in all scenarios.

One scenario, see Figures 4c and Figure 4f upper left corner, stands out, since both IANS and MPTCP show rather large performance penalties. Here, both access networks have a large latency (100 ms) and the downstream capacities are asymmetric (2/20 Mbit/s). Per se, using network 2 only is the best option. However, both IANS and MPTCP use network 1 as well, as its latency is lower than network 2's due to the WiFi overhead. By using network 1 IANS incurs extra cost due to TCP connection overhead and both IANS and MPTCP degrade performance due to the imposed congestion on network 1. Moreover, IANS cannot reuse existing TCP connections on a different network. Thus, in the future, IANS should account for connection establishment overhead and congestion even more.

Summary of systematic study: Our most important findings are: 1.) IANS improves Web performance the most for asymmetric scenarios and for symmetric scenarios with low downstream capacity. 2.) For asymmetric scenarios IANS often outperforms MPTCP. Indeed, MPTCP may introduce a performance penalty even compared to using a single network only. 3.) In symmetric scenarios, using MPTCP helps the most. Thus, IANS should take advantage of MPTCP for such cases.

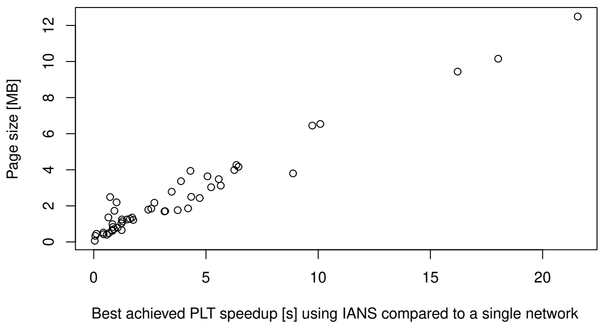


Fig. 5: Symmetric scenario with 10ms, 2Mbit/s.

C. Performance Benefits For Different Web Pages

Next, we take a closer look at Web page characteristics and check to which extent they correlate with the observed speedups. The most obvious characteristic is Web page size, i.e., the sum of all resource sizes from one page load. In our workload, Web page sizes range from 64 KB to 12.5 MB. Thus, Figure 5 shows a scatter plot of the Web page size vs. the best achieved PLT speedup for one of the symmetric scenarios (10 ms and 2 Mbit/s). Note, there is a strong correlation. Almost all points fall on the diagonal. Thus, the Pearson correlation coefficient with 0.978 is also close to the perfect one. Results for ATF are similar to PLT. For MOS we see some improvements as well but not a strict correlation, which is not surprising given that MOS uses a logarithmic scale. Still, IANS, e.g., improves "bad" MOS' (< 2.5) to "fair" (> 2.5) for 10% of the Web pages.

We find similar strong correlations between Web page size and PLT for almost all symmetric scenarios. Correlation decreases with larger downstream network capacities, as the best options for such scenarios is using a single network. When considering Web pages of different categories, see Section IV-B, we find that "Gaming" Web pages are often larger, so they show more significant speedups than, e.g., smaller Web pages from the "Search Engine" category.

We do not find strong correlations for other Web page characteristics including resource count and median resource size. Resource count is the number of successful HTTP requests as observed from the HAR file, which ranges from 10 to 314 in our workload, with a median of 73. Median resource size is the median of all resource sizes loaded for a single page, which ranges from 300 B to 79 KB in our workload. The largest Pearson correlation coefficients for resource count/median resource size are 0.517/0.541 for the symmetric scenario (10 ms and 2 Mbit/s).

For asymmetric scenarios, we see correlations between page size and PLT when the downstream capacity is low for both networks (Pearson coefficients greater than 0.8). Otherwise, there is no strong correlation with page size as the benefits of network capacity bundling decreases. There is also no strong correlation with resource count or median resource size.

Still, IANS shows improved performance, recall Section V-A. The reason is that IANS provides multiple opportunities for speedups: On the one hand, IANS provides benefits for large Web pages and those with many large resources. On the other hand, IANS realizes speedups for small pages with a few large resources and many small resources. Here, large resources benefit from the high downstream capacity network, while small resources benefit from the short latency network. Moreover, distributing resources avoids congestion on the lower latency network. This is where IANS shows an advantage over MPTCP.

IANS cannot speed up Web page loads where the page exclusively consists of small resources, since using a single short latency network while reusing the TCP connections is the best option. Moreover, IANS does not provide much benefit if the Web page consists of many small and few large resources, e.g., a single large resource. In such cases, scheduling the large resource on the large downstream capacity network may not always be beneficial due to its delay and the corresponding connection establishment overhead.

D. Performance In The Wild

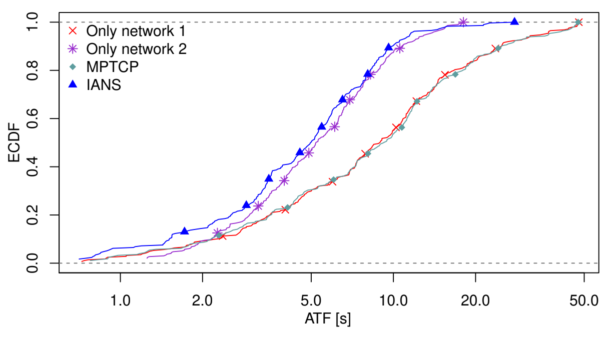
To confirm that the performance benefits do not just apply to fully controlled testbed settings with mirrored Web pages, we load the same pages from their servers "in the wild", recall Section IV-A.

Figure 6a shows the ECDF for the Above-The-Fold Times[14] (ATF) for the asymmetric network scenario (network 1: 10 ms and 2 Mbit, network 2: 100 ms and 20 Mbit) first discussed in Section V-A. As before IANS outperforms either of the two single networks as well as MPTCP. Indeed, MPTCP is much worse and close to using only the low capacity network for most Web pages. The reason is that most Web servers "in the wild" do not support MPTCP: Only three out the of 102 Web pages in our workload partially support MPTCP—we see at least one successfully established subflow with an MPTCP option. Note, even for these not all involved Web servers did support MPTCP. Therefore, while our testbed results show a meaningful comparison between IANS and MPTCP, the actual Web servers do not allow such a comparison. Without server-sided MPTCP support, MPTCP establishes the primary subflow over the lower latency network, i.e., network 1, and suffers from its low downstream capacity. Thus, the fact that MPTCP needs server-sided support limits the performance benefits that it can achieve in the wild. Since IANS does not need any server-sided support IANS can provide similar benefits in the wild as in the testbed.

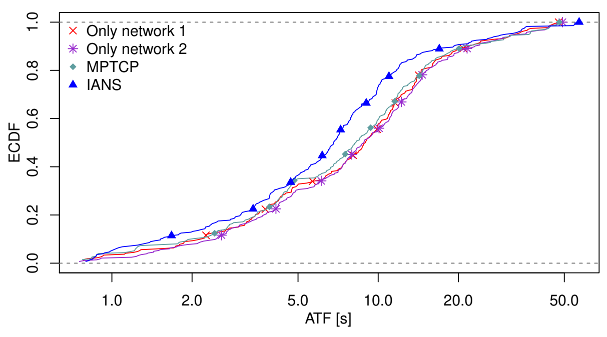
Figure 6b shows the ECDF for the ATF for a symmetric network scenario (10 ms and 2 Mbit/s). Again MPTCP does not provide any benefits compared to using a single network as most servers do not yet support MPTCP. Here, even though MPTCP outperforms IANS in the testbed, see Section V-B, IANS in the wild currently outperforms MPTCP (due to its limited deployment) as well as only using a single network.

In summary, our experiments using Web servers "in the wild" confirm IANS's performance benefits. Furthermore, they highlight that MPTCP, due to its limited deployment, often cannot provide the expected performance benefits in practice and may even degrade performance.

[14]PLT and MOS, again, show similar effects.



(a) Asymmetric scenario.



(b) Symmetric scenario.

Fig. 6: "In the wild": ECDF of Above-The-Fold Times (ATF)

VI. Related Work

Access Network Performance: Sommers et al. [25] compare WiFi and cellular performance between February and June 2011. They find performance varies widely between access technology and provider and that WiFi provides a higher throughput and a shorter latency in most cases. Between September 2013 and May 2014, Deng et al. [6] find that the assumption that WiFi outperforms cellular is no longer true. Our work builds on these findings, as there is no single obvious "best" access network anymore.

Offloading: Shifting traffic from the cellular network to the WiFi—offloading—has gotten a lot of attention both in the research community, e.g., [17], [16], [1], as well as in industry, e.g., [15], [4]. For a survey on offloading we refer to Maallawi et al. [17]. Several studies demonstrate the potential benefits of augmenting mobile 3G networks with Wifi, e.g., Lee et al. [16] and Balasubramanian et al. [1]. In contrast, Rossi et al. [21] find that 3G networks can augment WiFi performance, which they call onloading. Wiethölter et al. [29] compare strategies to determine which mobile subscribers to offload based on Received Signal Strength or a combination of throughput efficiency and WiFi channel load. Galdgil et al. [10] consider offloading strategies based on the application traffic characteristics, such as real-time requirements. Our work complements this work by dynamically off- or onloading traffic based on both the current network performance and application traffic characteristics.

Multipath Protocols: In addition to shifting traffic from cellular to WiFi and vice versa, MPTCP [9] can aggregate both networks. Performance studies by Chen et al. [3] find that MPTCP provides benefits for large flows, but not for small flows. Deng et al. [6] observe that MPTCP may penalize short flows. Han et al. [11] study Web performance over MPTCP. They see cases in which HTTP over MPTCP performs even worse than single-path TCP. Furthermore, in their results, the lower latency network usually dominates given similar capacity and loss. Our results are in line with these performance studies while offering IANS as an alternative. In order to overcome deployment limitations of MPTCP, Nikravesh et al. [19] propose a client-only HTTP/2-based multipath solution. While tackling a similar problem as IANS, their study is limited to transferring single files, while we load Web pages.

Multipath Support in Systems: To make access network selection usable on end-user devices, there has been work on network layer multipath [20] and multi-access connectivity [24]. While this allows using multiple access networks, it does not facilitate informed decisions based on input from the application. Application input requires enhanced networking APIs. Early efforts include Intentional networking [12], which lets applications specify network requirements via an extended Socket API. However, they take a per-packet approach for a specific use case, while IANS is more general and uses a per-flow approach. Moreover, they imply guarantees while IANS suggests best-effort. More recently, NEAT [14] is a transport protocol-independent API which also allows to select different paths, thus, different access networks. While their work focuses on transport protocol selection, our work focuses on access network selection. Both the NEAT and the Socket Intents prototype serve as input to the TAPS API [28] standardization effort in the IETF.

VII. Summary and Future Work

We leverage Socket Intents to achieve IANS. Informed by Socket Intents, i.e., the Size to be Received of a Web resource, we select the most suitable access network to load the resource. We implement IANS within the Socket Intents prototype and evaluate its benefits for Web performance within a testbed and using Web servers "in the wild". Our holistic study reveals that IANS improves Above-The-Fold Times by between 500 and 1000 ms in median in asymmetric network scenarios and in network scenarios with low downstream capacity. For asymmetric network scenarios, IANS outperforms MPTCP since it can avoid self-induced congestion on the low capacity network. For symmetric network scenarios, MPTCP may be able to use the available capacity more efficiently, but it may not be available "in the wild". Therefore, in future work, an advanced IANS Policy should combine the benefits of IANS and MPTCP. Moreover, the IANS Policy should be refined to recognize cases in which a network is congested so that loading resources over it does not provide any benefit. Furthermore, future work on IANS has to explore additional optimizations, such as distributing resource loads across multiple CDN nodes serving the same content. Finally, future work should evaluate the benefits of IANS for other applications, such as video streaming, uploading content, and instant messaging.

References

[1] Balasubramanian, A., Mahajan, R., and Venkataramani, A. Augmenting mobile 3g using wifi. In *ACM MobiSys* (2010), pp. 209–222.

[2] Bocchi, E., De Cicco, L., and Rossi, D. Measuring the quality of experience of web users. *ACM CCR 46*, 4 (2016), 8–13.

[3] Chen, Y.-C., Lim, Y.-s., Gibbens, R. J., Nahum, E. M., Khalili, R., and Towsley, D. A measurement-based study of multipath tcp performance over wireless networks. In *ACM IMC* (2013), ACM, pp. 455–468.

[4] Cisco Systems, Inc. Architecture for mobile data offload over wi-fi access networks (whitepaper), 2012.

[5] da Hora, D. N., Asrese, A. S., Christophides, V., Teixeira, R., and Rossi, D. Narrowing the gap between qos metrics and web qoe using above-the-fold metrics. In *International Conference on Passive and Active Network Measurement* (2018), Springer, pp. 31–43.

[6] Deng, S., Netravali, R., Sivaraman, A., and Balakrishnan, H. Wifi, lte, or both?: Measuring multi-homed wireless internet performance. In *ACM IMC* (2014), ACM, pp. 181–194.

[7] Enghardt, T., Tiesel, P. S., and Feldmann, A. Metrics for access network selection. In *Proceedings of the Applied Networking Research Workshop* (New York, NY, USA, 2018), ANRW '18, pp. 67–73.

[8] Enghardt, T., Zinner, T., and Feldmann, A. Web performance pitfalls. In *PAM* (2019), PWS Publishing Company, pp. 286–303.

[9] Ford, A., Raiciu, C., Handley, M., and Bonaventure, O. TCP Extensions for Multipath Operation with Multiple Addresses. RFC 6824 (Experimental), Jan 2013.

[10] Gadgil, S., Ranjan, S., Joshi, D., Mehta, M., Akhtar, N., and Karandikar, A. Performance evaluation and viability of ifom in heterogeneous lte—wlan network. In *2015 IEEE Wireless Communications and Networking Conference (WCNC)* (2015), IEEE, pp. 1524–1529.

[11] Han, B., Qian, F., Hao, S., and Ji, L. An anatomy of mobile web performance over multipath tcp. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies* (2015), ACM, p. 5.

[12] Higgins, B. D., Reda, A., Alperovich, T., Flinn, J., uli, T. J. G., Noble, B., and Watson, D. Intentional networking: opportunistic exploitation of mobile network diversity. In *ACM MobiCom* (2010), ACM, pp. 73–84.

[13] Hossfeld, T., Metzger, F., and Rossi, D. Speed index: Relating the industrial standard for user perceived web performance to web qoe. In *IEEE International Conference on Quality of Multimedia Experience* (2018).

[14] Khademi, N., Ros, D., Welzl, M., Bozakov, Z., Brunstrom, A., Fairhurst, G., Grinnemo, K.-J., Hayes, D., Hurtig, P., Jones, T., et al. Neat: a platform-and protocol-independent internet transport api. *IEEE Communications Magazine 55*, 6 (2017), 46–54.

[15] Lee, J., Yi, Y., Chong, S., and Jin, Y. Economics of wifi offloading: Trading delay for cellular capacity. *Wireless Communications, IEEE Transactions on 13*, 3 (2014), 1540–1554.

[16] Lee, K., Lee, J., Yi, Y., Rhee, I., and Chong, S. Mobile data offloading: how much can wifi deliver? In *ACM CoNEXT* (2010).

[17] Maallawi, R., Agoulmine, N., Radier, B., and Meriem, T. B. A comprehensive survey on offload techniques and management in wireless access and core networks. *IEEE Communications Surveys & Tutorials 17*, 3 (2015), 1582–1604.

[18] Netravali, R., Sivaraman, A., Das, S., Goyal, A., Winstein, K., Mickens, J., and Balakrishnan, H. Mahimahi: Accurate record-and-replay for http. In *2015 USENIX Annual Technical Conference (USENIX ATC 15)* (2015), pp. 417–429.

[19] Nikravesh, A., Guo, Y., Zhu, X., Qian, F., and Mao, Z. Mp-h2: A client-only multipath solution for http/2: A client-only multipath solution for http/2.

[20] Qadir, J., Ali, A., Yau, K.-L. A., Sathiaseelan, A., and Crowcroft, J. Exploiting the power of multiplicity: a holistic survey of network-layer multipath. *IEEE Communications Surveys & Tutorials 17*, 4 (2015), 2176–2213.

[21] Rossi, C., Vallina-Rodriguez, N., Erramilli, V., Grunenberger, Y., Gyarmati, L., Laoutaris, N., Stanojevic, R., Papagiannaki, K., and Rodriguez, P. 3gol: Power-boosting adsl using 3g onloading. In *ACM CoNEXT* (2013), ACM, pp. 187–198.

[22] Scheitle, Q., Hohlfeld, O., Gamba, J., Jelten, J., Zimmermann, T., Strowes, S. D., and Vallina-Rodriguez, N. A long way to the top: significance, structure, and stability of internet top lists. In *ACM IMC* (2018), pp. 478–493.

[23] Schmidt, P. S., Enghardt, T., Khalili, R., and Feldmann, A. Socket intents: Leveraging application awareness for multi-access connectivity. In *ACM CoNEXT* (New York, NY, USA, 2013), pp. 295–300.

[24] Schmidt, P. S., Merz, R., and Feldmann, A. A first look at multi-access connectivity for mobile networking. In *ACM workshop on Capacity sharing* (2012), ACM, pp. 9–14.

[25] Sommers, J., and Barford, P. Cell vs. wifi: On the performance of metro area mobile connections. In *ACM IMC* (2012).

[26] Sommers, J., Kim, H., Barford, P., and Barford, P. Harpoon: a flow-level traffic generator for router and network tests. In *ACM SIGMETRICS Performance Evaluation Review* (2004), vol. 32, ACM, pp. 392–392.

[27] Tiesel, P., Enghardt, T., and Feldmann, A. Socket intents. Internet-Draft (work in progress) draft-tiesel-taps-socketintents-01, IETF, 10 2017.

[28] Trammell, B., Welzl, M., Enghardt, T., Fairhust, G., Kuehlewind, M., Perkins, C., Tiesel, P., and Wood, C. An abstract application layer interface to transport services (work in progress). Internet Draft (work in progress) draft-ietf-taps-interface-03, IETF, March 2019.

[29] Wiethölter, S., Emmelmann, M., Andersson, R., and Wolisz, A. Performance evaluation of selection schemes for offloading traffic to ieee 802.11 hotspots. In *2012 IEEE International Conference on Communications (ICC)* (2012), IEEE, pp. 5423–5428.